reads a MRPF, a pane row, or a MRPI for filters pane in step 292. The filter is then executed with the changes in step 293. The method then test whether there are other MRPI or PRT affected by the filter changes and then returns to step 292 to process them if they are present.

Referring now to FIGS. 15A and 15B, the preferred method for executing a function for a change to the MRPI, as used in FIG. 17 below, is described. In step 300, the images of the MRPIs before and after the changes to the MRPI are read. The present method preferably saves images of the MRPI whenever anything is changed. These images are saved until the changes have been processed. In step 301, the MRPF of the changed MRPI is read. Then in step 302, the calculated field of the before and after images are compared. In step 303, the method tests whether the value for each calculated image has changed. If there is no change, the method repeats step 302 to get the next calculated field. However, if there is a change, the method moves to step 304. In step 304, the function(s) that use this calculated field are retrieved. In step 305, the method tests whether any of the functions using this calculated field are group by functions. If a group by function is involved, the method first reads the MRPI's parent MRPI where the calculated amount is stored in step 307. Then in step 308, the group by value is recalculated. If a group by function is not involved, the method recalculates the function for this MRPI in step 306. Next in step 309, the method tests whether this is the last function affected by the change. If more functions are affected, the method returns to step 304 to process those other functions. Otherwise, the method continues to step 310 to determine if this is the last calculated field that has been modified. If there are more calculated fields to process the method returns to step 302. If not the process is complete and all the values are function value.

FIGS. 16A and 16B illustrate a preferred method for executing a filter for a new or changed set of DDPs. DD functions work almost identically to MRT functions. The calculation logic is the same except DDPs correspond to MRPIs. Panes correspond to MRPFs, and top pane DDPs correspond to MRIs. MRT functions are always executed prior to DD functions. Also, if a pane does not include in itself or its parents, the lead PRT for its own lowest level MRPFs, it must be summarized. Summarization is just the process of grouping then the pane and its parent panes. Calculated fields from the pane's MRPFs must be grouped by one of the standard group by operands (See Appendix A). The user specifies the group by operand when he/she defines the DD.

The method starts in step 320 with the DD definition being read. The present invention processes each dynamic document instance (DDI) separately beginning with the lowest level panes in the pane hierarchy and working upward.

In step 231, the top-level pane DDI is read and the variable PASS LEVEL is set equal to zero. In step 322, the method reads the lowest unprocessed pane hierarchy level for this pass. In step 323, the method reads the pane's functions. Then in step 324, the method reads other pane's group by function(s) that use this level. Group by functions are executed with the pane they summarize not the pane where the result is stored. The DDPs for this pane are then read in step 325. Then in step 326, the function for this pane are executed. The group by accumulators are then updated in step 327. Next in step 328, the system determines whether the DDP being used to execute the functions is the last DDP for this pane. If not, the method returns to step 325, otherwise the process continues in step 329. In step 329, the group by accumulator values are added to the DDI. Then in step 330, the method test whether this is the last pane for this DD 20. If not, the process loops to step 322 to process any unprocessed panes, otherwise the process continues in step 331. In step 331, the method determines if this is the last pass. If it is not the last pass, then 1 is added to the PASS LEVEL in step 332 and the method returns to step 322 for further processing. After each pass of all the DDI, the method executes the functions with the next highest pass sequence number until all functions are executed. However, if it is the last pass, the method continues in step 332 where the process tests whether this is the last top DDI to perform functions for. If not, the process returns to step 321, otherwise the execution of the functions for the DD is complete.

All changes to any data is entered by the user through DDs or fetches from a production data source. All changes can affect multiple DDs and MRTs. For example, if the user changes a customer's discount percentage and multiple MRTs use that field in their calculations, the present invention must update many MRPIs and DDPs for that one change. Therefore, when the user enters a change, the present invention first updates the affected PRT and then the MRPIs and finally the DDPs that use that PRT. The updates to the MRPIs have been described above with reference to FIGS. 4 and 5.

FIG. 17 illustrates the preferred method for updating a DD pane's dynamic documents pointers (DDPs) and the displayed records for data changes. Users enter data changes through any of a MR's DDs. Newly imported XML documents also create data changes. The preferred method identifies data changes and ensures that all affected DDP records are either updated with the new values or recreated. (The method for filtering and displaying DD pane data is described in FIG. 18.) In step 340, the method maintains PRIs for data enter through a DD or imported as XML documents. The changed values can be changes to existing PRI field values including field values that are also hooks to other PRIs, as well as changes that are also new PRIs. In step 341, the method maintains the MPRIs for the changes PRIs as described in FIG. 10 for new PRIs and FIG. 11 for existing PRIs.

Next the method checks every DD that is currently activated to determine the changes, if any to make to its DDP and the displayed records. In step 342, the method tests whether the PRT is used by the DD; if not used no further processing is done on that DD. If the DD does use the PRT, step 343 checks to see if any hook fields were changed. If so step 344 deletes all the DDP records that references the changed PRT. The method also deletes all DDP records in DD panes lower in the DD pane hierarchy than that of the affected DDP. Step 344 then it recreates the DDP records as described in FIG. 9 and proceeds to step 347. When a hook field was not changed, step 345 updates the DDP record field values with the new PRI field values and runs the DD filters and functions on the changed record. Then in step 346, the method runs the filters and functions for all records in DD panes that are lower in the DD pane hierarchy and also reference the changed DDP record. In step 347, the method runs the filters and functions for all records that are higher in the DD pane hierarchy and reference any DDP record changed by steps 344, 345, or 346. When step 348 determines the method changed any field value in a DDP record used by the sort rules, it sorts the DD pane in step 349. Finally the method redisplays all DDP records that it modified or resorted.

As an alternate method, steps that would determine which DDP records were affected by the change in a hook value could replace step 344. This alternate method would not reconstruct all the DDP records, but would modify, delete, and insert only the records required by the field value changes. This method could be more efficient when the ratio of changed DDP records to the total number of records is very low.